(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **11003105 A**

(43) Date of publication of application: **06.01.99**

(51) Int. Cl **G05B 19/05**

(21) Application number: **09152151**

(22) Date of filing: **10.06.97**

(71) Applicant: **DENSO CORP**

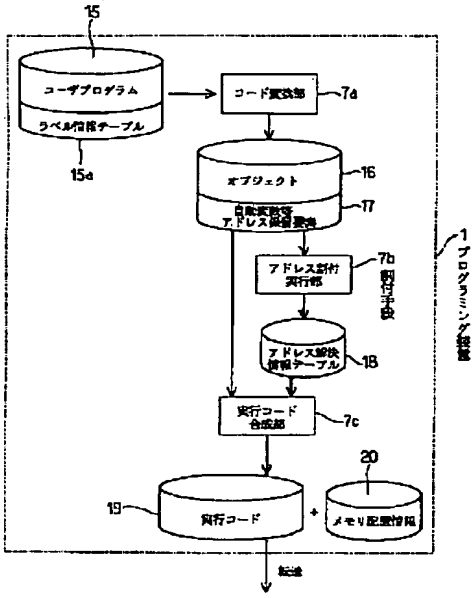
(72) Inventor: **ITO MARIKO**
SATO MASAHIKO

(54) PROGRAMMING DEVICE FOR PROGRAMMABLE CONTROLLER

(57) Abstract:

PROBLEM TO BE SOLVED: To provide a programming device for programmable controller with which a user program for executing complicated arithmetic or conditional processing can be easily described, comprehension is facilitated and security of data is guaranteed.

SOLUTION: A source file 15 of program is converted to an object file 16 by a code converting part 7a and the automatic variable, of name, to which a prefix 'Ud' is added, in the program is processed as an address holding element 17 for which a real address is not fixed yet. The real address of memory area on the side of programmable controller is assigned to the address holding element 17 by an address assignment execution part 7b, and the assigned result is generated as an address solution information table 18. Information in the address solution information table 18 is synthesized with the object code in the object file 16 by an execution code synthesizing part 7c, and an execution code file 19 is generated together with memory arrangement information 20.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－3105

(43)公開日　平成11年(1999)１月６日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０５Ｂ　19/05		Ｇ０５Ｂ　19/05	Ｂ
			Ｆ

審査請求　未請求　請求項の数５　ＯＬ　（全 11 頁）

(21)出願番号　特願平9－152151

(22)出願日　平成９年(1997)６月10日

(71)出願人　000004260
株式会社デンソー
愛知県刈谷市昭和町１丁目１番地

(72)発明者　伊藤　万里子
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内

(72)発明者　佐藤　雅彦
愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内

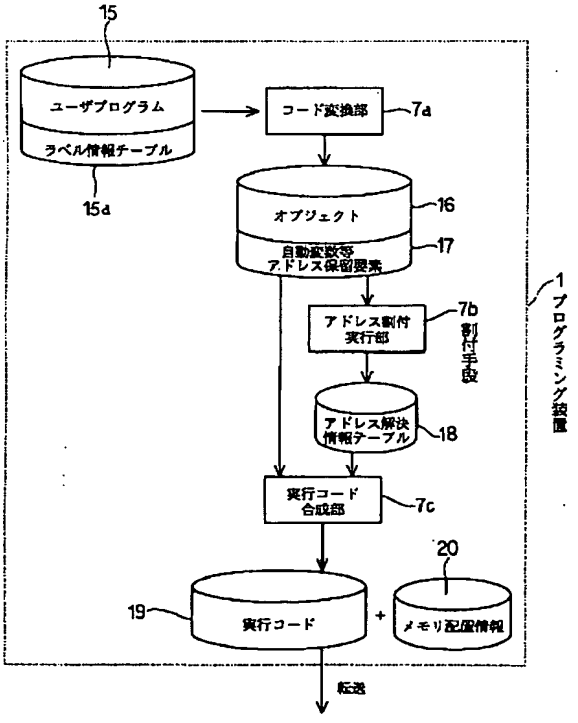
(74)代理人　弁理士　佐藤　強

(54)【発明の名称】　プログラマブルコントローラのプログラミング装置

(57)【要約】

【課題】　複雑な演算または条件処理を行なうユーザプログラムを容易に記述でき、且つ、理解が容易でデータの安全性が保証されるプログラマブルコントローラのプログラミング装置を提供する。

【解決手段】　プログラムのソースファイル１５は、コード変換部７ａによりオブジェクトファイル１６に変換され、プログラム中で接頭語“Ｕｄ”が付された名称の自動変数は、実アドレスが未確定のアドレス保留要素１７として処理される。アドレス保留要素１７は、アドレス割付実行部７ｂによってプログラマブルコントローラ側のメモリエリアの実アドレスが割付けられ、その割付け結果は、アドレス解決情報テーブル１８として生成される。アドレス解決情報テーブル１８の情報は、実行コード合成部７ｃによってオブジェクトファイル１６のオブジェクトコードと合成されて、実行コードファイル１９とメモリ配置情報２０とが生成される。



【特許請求の範囲】
【請求項１】　ユーザプログラムを、プログラマブルコントローラが実行可能な形式に変換するプログラム変換手段を備えたプログラマブルコントローラのプログラミング装置において、
前記プログラム変換手段は、前記ユーザプログラム中において、ユーザにより特定の名称が付与された変数が使用されていると、前記ユーザプログラムをプログラマブルコントローラが実行可能な形式に変換する際に、前記変数を介して渡されるデータの格納領域をプログラマブルコントローラ側に設定される割付領域中に割付けるように処理する割付手段を備えていることを特徴とするプログラマブルコントローラのプログラミング装置。
【請求項２】　前記割付領域は、前記変数を介して渡されるデータのみが格納可能に設定されることを特徴とする請求項１記載のプログラマブルコントローラのプログラミング装置。
【請求項３】　前記割付領域の容量がユーザによって指定されている場合は、前記指定に応じて前記割付領域の容量を確保する領域設定手段を備えたことを特徴とする請求項１または２記載のプログラマブルコントローラのプログラミング装置。
【請求項４】　前記割付手段は、前記変数をオペランドとする命令の種類に応じて当該変数の属性を判断し、前記属性に応じた割付領域に前記変数を介して渡されるデータの格納領域を割付けるように処理することを特徴とする請求項１乃至３の何れかに記載のプログラマブルコントローラのプログラミング装置。
【請求項５】　前記割付手段は、新たな変数の割付け処理を行う時点で前記割付領域中に空き領域が存在しない場合には、前記割付領域中に既に割付けられている変数の内、前記新たな変数が使用されているステップ以降のユーザプログラムでは使用されない変数の領域に、前記新たな変数を介して渡されるデータの格納領域を割付けることを特徴とする請求項１乃至４の何れかに記載のプログラマブルコントローラのプログラミング装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、ユーザプログラムをプログラマブルコントローラが実行可能な形式に変換するプログラム変換手段を備えたプログラマブルコントローラのプログラミング装置に関する。
【０００２】
【発明が解決しようとする課題】近年、プログラマブルコントローラ（以下、ＰＣと称す）の制御対象となる設備は大規模になると共に複雑化しており、それに伴って、ＰＣのユーザプログラム（制御プログラム）及びそれに付随するデータも大容量化してきている。
【０００３】特にユーザプログラムの記述に用いられるラダー言語によって複雑な演算を行なう場合には、演算を１ステップずつ分割して順に行なわねばならず、演算の途中結果などのデータを一時的にメモリに格納させる処理が頻繁に発生する。
【０００４】また、多数の入力リレーによる工程完了条件を判定する場合も同様であり、前記工程完了条件を１つのラングにおいて記述可能なステップ数の範囲に分割してプログラムを記述し、各ラング毎に記述された入力リレーによる前記条件の一部が成立した場合は、その情報を一旦補助リレー等に格納させて、複数の補助リレーの情報を元にして前記条件の全てが成立したか否かを判定する必要がある。
【０００５】その際、一時的に演算データやリレー情報をＰＣのメモリに格納させるアドレスは、ユーザ自身がその都度メモリの使用可能な領域を検索した上で指定する必要があった。従って、ユーザプログラムの大規模化に伴って、上記作業が非常に時間を要するようになり、プログラムの作成効率を低下させているという問題があった。加えて、メモリの既に使用中の領域を誤って指定する危険性があり、不具合の発生頻度が上昇しているという問題も生じていた。
【０００６】本発明は上記事情に鑑みてなされたものであり、その目的は、複雑な演算または条件処理を行なうユーザプログラムを容易に記述でき、且つ、理解が容易でデータの安全性が保証されるプログラマブルコントローラのプログラミング装置を提供することにある。
【０００７】
【課題を解決するための手段】請求項１記載のプログラマブルコントローラのプログラミング装置によれば、ユーザは、プログラミングを行なう際に、変数に対するアドレスの割付処理を一切意識する必要がなく、特定の名称を付与した変数をユーザプログラム中で使用すれば、その変数を介して渡されるデータの格納領域は、プログラム変換手段が有する割付手段によってプログラマブルコントローラ側に設定される割付領域中に割付けられる。従って、ユーザプログラムの作成が容易になり、作成を効率良く行なうことができる。また、ユーザが付与した変数名を使用することによって作成後のユーザプログラムの理解も容易となる。
【０００８】請求項２記載のプログラマブルコントローラのプログラミング装置によれば、割付領域は、変数を介して渡されるデータのみが格納可能に設定されるので、ユーザプログラムにおいて、不用意に割付領域に対応するアドレスを記述することに起因して割付領域に格納されているデータを破壊してしまうことがなく、データの保護を確実に行なうことができる。
【０００９】請求項３記載のプログラマブルコントローラのプログラミング装置によれば、ユーザが、作成するプログラムの内容に応じて必要となる割付領域の容量を指定すれば、その指定に応じて割付領域の容量が確保されるので、ユーザプログラムの作成をより柔軟に行なう

ことができる。

【００１０】請求項４記載のプログラマブルコントローラのプログラミング装置によれば、ユーザは、ユーザプログラムにおいて一々変数の属性（例えば、ワード属性かリレー属性か）を指定せずとも、割付手段は、命令の種類に応じて変数の属性を判断して割付けを行なうので、ユーザプログラムの作成をより容易且つ効率的に行なうことができる。

【００１１】請求項５記載のプログラマブルコントローラのプログラミング装置によれば、割付手段は、割付領域中に新たな変数の割付処理を行うための空き領域が存在しない場合には、既に割付けられている変数の領域の内、新たな変数が使用されているステップ以降のユーザプログラムでは使用されない変数の領域に、新たな変数を介して渡されるデータの格納領域を割付けるので、割付領域を効率的に利用して、変数を使用するために必要な容量を低減することができる。

【００１２】

【発明の実施の形態】以下、本発明の一実施例について図面を参照して説明する。図７及び図８は、プログラマブルコントローラ及びそのプログラミング装置の構成を示す機能ブロック図である。図８において、プログラミング装置１は、例えばパーソナルコンピュータによって構成されており、キーボード２，マイクロコンピュータを中心とする制御部３，ディスプレイ４などからなるものである。

【００１３】制御部３には、プログラマブルコントローラ（以下、ＰＣと称す。図７参照）５に実行させるユーザプログラム（以下、単にプログラムと称す）を入力，修正するためのソフトウエアであるエディタ６，後述するように割付領域の容量を設定する領域設定部（領域設定手段）６ａ，入力されたプログラムの変換処理などを行う変換処理部（プログラム変換手段）７が、ハードディスク８から読み出されて、図示しない主記憶エリアたるＲＡＭ上に常駐しているものとする。

【００１４】プログラミング装置１は、更に、ＰＣ５との間でデータ転送を行うための通信用インターフェイス９などを有している。通信用インターフェイス９は、ＲＳ－２３２Ｃ若しくはＲＳ－４８５などのシリアル通信方式によって、通信用ケーブル１０を介してデータ転送を行うようになっている。

【００１５】一方、図７に示すように、ＰＣ５側は、ＣＰＵ１１，そのＣＰＵ１１にシステムバスを介して接続されているＲＯＭ１２及びＲＡＭ１３，プログラミング装置１側の通信用インターフェイス９と通信ケーブル１０を介して接続され、ＣＰＵ１１とも通信用バスで接続されている通信用インターフェイス１４などから構成されている。

【００１６】ＲＯＭ１２には、ＣＰＵ１１の基本的な制御を行うプログラムであるファームウエアが記憶されており、ＲＡＭ１３には、プログラミング装置１側から送信されるユーザプログラム及びデータが記憶されるようになっている。また、ＣＰＵ１１には、Ｉ／Ｏバスを介して図示しない他のＩ／Ｏ装置が接続されている。

【００１７】次に、本実施例の作用について図１乃至図６をも参照して説明する。作業者（ユーザ）は、プログラミング装置１のエディタ６を起動すると、キーボード２を操作して、ディスプレイ４の表示を見ながらラダー言語によるユーザプログラム（以下、単にプログラムと称す）を作成する。

【００１８】作業者は、先ず、プログラムを記述する前に、自動変数（変数）の割付領域のサイズ（容量）を指定する。例えば、自動変数リレー割付領域（以下、リレー割付領域と称す），自動変数ワード割付領域（以下、ワード割付領域と称す）の容量を夫々２５６ワードずつ確保するように指定する。尚、１ワードは２バイトを意味する。

【００１９】すると、領域設定部６ａは、図２に一例を示すＰＣ５のユーザメモリマップに割付領域の確保を行う。このメモリマップ上では、００００Ｈ番地から１０００Ｈ番地まではリレー領域、１０００Ｈ番地から８０００Ｈ番地まではワード領域として大別されている。リレー領域はビット指定が可能な領域である。

【００２０】そして、このメモリマップの０Ｆ００Ｈ番地から０ＦＦＦＨ番地までの２５６ワードの領域をリレー割付領域として、７Ｆ００Ｈ番地から７ＦＦＦＨ番地までの２５６ワードの領域をワード割付領域として確保する。また、１０００Ｈ番地から２ＦＦＦＨ番地までは、データメモリ領域（Ｄ）としてシステム上予め設定されている。

【００２１】尚、リレー及びワード割付領域は、何れも自動変数を介してのみデータの格納が可能な領域として設定されており、ユーザプログラム中では、これらの領域に相当するアドレスを直接記述することはできないようになっている。

【００２２】次に、作業者は、図３に示すように、自動変数を使用してプログラムを記述する。この図３に示すプログラムの意味は、以下の通りである。

①工程Ａ完了条件

X0:0,X0:1,X0:2,X0:3,X0:4:ON or X0:0,X0:6,X0:7:ON X0:5:OFF

②工程Ｂ完了条件

X2:0,X2:1,X2:2,X2:3,X2:4:ON or X2:2,X2:3,X2:4:ON X3:0,X3:1:OFF

③工程Ａ及び工程Ｂが共に完了すると、以下の演算を行なう。

Ｄ１０＝（Ｄ０＋Ｄ１）－（Ｄ２＊Ｄ３）

…

【００２３】この時、作業者は、自動変数を使用する場合は、識別子“Ｕｄ”を接頭語とする適当な変数の名称

を付与してプログラム中で記述すれば良い。例えば、第１ラングの“Ｕｄ工程Ａ完了”は、①の工程Ａの完了条件が成立した時にデータ“１”がセットされる自動変数である。即ち、工程Ａ及び工程Ｂが共に完了することにより、“Ｕｄ工程Ａ完了”及び“Ｕｄ工程Ｂ完了”に共に“１”がセットされると、第３ラングの演算が実行される。

【００２４】また、X0:0,X0:1,X0:2, …は入力リレーのアドレスを指定する記述子、Ｄ０～Ｄ３，Ｄ１０は、データメモリ領域のアドレスを指定するためにシステム上予め用意されている記述子である。例えば、X0:0は、入力リレー領域（Ｘ）の０番地の第０ビット（即ち、図３に示すメモリマップにおける０００００：０Ｈ番地の第０ビット）を表し、“Ｄ０”は、データメモリ領域（Ｄ）の０番地（同１０００Ｈ番地）を表している。

【００２５】また、“Ｕｄ演算結果１”及び“Ｕｄ演算結果２”は、③の演算の途中結果が格納される自動変数である。ラダープログラムの制約によって、③の演算を１つの式で記述して一気に実行させることができないため、第３ラングでは、先ず③の右辺第１項を演算してその結果を“Ｕｄ演算結果１”に格納し、次のステップで同第２項を演算してその結果を“Ｕｄ演算結果２”に格納する。そして、更に次のステップにおいて、“Ｕｄ演算結果１”の内容から“Ｕｄ演算結果２”の内容を減算することにより③の演算が完了し、その演算結果が“Ｄ１０”に格納されるようになっている。

【００２６】以上のようにプログラムが記述されると、プログラミング装置１においては、制御部３の変換処理部７によって、図１に示すような手順に従いプログラムが処理される。即ち、プログラムのソースファイル１５は、コード変換部７ａによってオブジェクト（機械語コード）ファイル１６に変換される。

【００２７】また、プログラム中で使用されているラベル（自動変数を含む）に関する情報は、プログラムの入力が確定した時点でラベル情報テーブル１５ａとして生成されており、そのラベル情報テーブル１５ａ中にある自動変数については、コード変換部７ａによって実アドレスが未確定のアドレス保留要素１７として処理される。

【００２８】尚、X0:0,X0:1,X0:2, …や、Ｄ０～Ｄ３などのシステム上予め用意されている記述子のアドレスは、コード変換部７ａによって、夫々対応するリレー領域、データメモリ領域の番地に変換されるようになっている。

【００２９】次に、オブジェクトファイル１６のアドレス保留要素１７部分は、アドレス割付実行部（割付手段）７ｂにおいて、ＰＣ５側のメモリエリア（ＲＡＭ１３）の実アドレスが割付けられ、その割付結果は、アドレス解決情報テーブル１８として生成される。そして、そのアドレス解決情報テーブル１８の情報は、実行コード合成部７ｃによってオブジェクトファイル１６のオブジェクトコードと合成されて、実行コードファイル１９と自動変数等に関するメモリ配置情報２０とが生成される。

【００３０】また、アドレス割付実行部７ｂは、変数の接頭語が“Ｕｄ”である場合は、図５に示すフローチャートに従ってアドレス保留要素１７に対してアドレスの割付け処理を行う。先ず、「メモリ属性はリレー？」の判断ステップＳ１において、命令パラメータのメモリ属性がリレーであるか否かを判断する。

【００３１】例えば、命令が“ＳＥＴ”のようにリレーを対象とするものであれば、そのオペランドに記述されている自動変数“Ｕｄ工程完了Ａ”のメモリ属性はリレーである。斯様な場合、アドレス割付実行部７ｂは「ＹＥＳ」と判断して、「リレー割付領域の空き領域検索」の処理ステップＳ２に移行する。また、命令が“＋”，“－”，“＊”等の演算子であればメモリ属性はワードであり、斯様な場合は「ＮＯ」と判断して「ワード割付領域の空き領域検索」の処理ステップＳ７に移行する。

【００３２】処理ステップＳ２において、アドレス割付実行部７ｂは、リレー割付領域を検索すると、次の「空き領域有り？」の判断ステップＳ３に移行して、空き領域が存在するか否かを判断する。空き領域が存在し「ＹＥＳ」と判断すると、「メモリ割付」の処理ステップＳ４に移行して、リレー割付領域に対するアドレスの割付を実行する。

【００３３】例えば、自動変数“Ｕｄ工程Ａ完了”，“Ｕｄ工程Ｂ完了”のアドレスは、領域が全て未使用の状態であれば、その先頭から０Ｆ００Ｈ番地の第０ビット（０Ｆ００Ｈ：０），第１ビット（０Ｆ００Ｈ：１）に夫々割付けられる。アドレスを割付けると、処理を終了する。

【００３４】また、判断ステップＳ３において、リレー割付領域に空きがなく、アドレス割付実行部７ｂが「ＮＯ」と判断すると、「非競合領域検索」の処理ステップＳ５に移行する。処理ステップＳ５において、アドレス割付実行部７ｂは、リレー割付領域の既に割付けが行われている領域の内、その時点で以降は不要となる自動変数に対応する領域、即ち、非競合領域を検索する。そして、次の「非競合領域有り？」の判断ステップＳ６において、非競合領域の有無を判断する。

【００３５】ここで、図３に示したプログラムの後が、図６に示すように続くものとした場合を考える。そして、第４ラングにおいて、“Ｕｄ工程Ｃ完了”という新たな自動変数をリレー割付領域に割付けようとした場合に空き領域がなければ、アドレス割付実行部７ｂは、既に使用されておりメモリ属性がリレーである自動変数のスコープ（有効範囲）をチェックして非競合領域を検索する。

【００３６】図６において、右端部に矢印で示している

のが各変数のスコープである。第４ラングの時点では、第１ラングで使用されている自動変数“Ｕｄ工程Ａ完了”のスコープは次の第５ラングまでであり、アドレス０Ｆ００Ｈ番地の第０ビットは競合領域であるが、第２ラングで使用されている自動変数“Ｕｄ工程Ｂ完了”のスコープは第３ラングまでであり、アドレス０Ｆ００Ｈ番地の第１ビットは非競合領域となっている。

【００３７】この場合、アドレス割付実行部７ｂは、判断ステップＳ６で「ＹＥＳ」と判断してステップＳ４に移行して、“Ｕｄ工程Ｃ完了”のアドレスを０Ｆ００Ｈ番地の第１ビットに割付ける。

【００３８】一方、判断ステップＳ６において、非競合領域が存在せず「ＮＯ」と判断した場合は、「割付領域不足をユーザに報知」の処理ステップＳ１０に移行して、アドレス割付実行部７ｂは、ディスプレイ４にエラーメッセージを出力させるなどして、割付領域が不足していることをユーザに報知すると処理を終了する。

【００３９】この場合、作業者は、領域設定部６ａによって、リレー割付領域の容量をより多く確保するように設定をやり直してから、再度プログラムの変換を実行させるようにして対応する。

【００４０】また、メモリ属性がワードである自動変数については、ステップＳ７～Ｓ９において、ワード割付領域に対して同様の割付け処理が行われる。例えば、第３，ラング使用されている自動変数“Ｕｄ演算結果１”，“Ｕｄ演算結果２”は、ワード割付領域である７Ｆ００Ｈ番地，７Ｆ０１Ｈ番地に夫々割付けられる。

【００４１】以上のようにアドレスの割付けが行われた結果、各命令のオペランドの実アドレスは図４に示すように設定され、上述のようにアドレス解決情報テーブル１８として生成され、更に、オブジェクトファイル１６のオブジェクトコードと合成されることにより、実行コードファイル１９及び自動変数と実アドレスとの対応を示すメモリ配置情報２０とが生成される。

【００４２】生成された実行コードファイル１９とメモリ配置情報２０とは、プログラミング装置１に例えば“セーブ”コマンドが入力されるとハードディスク８に書き込まれて記憶され、また、“送信”コマンドが入力されると、通信用インターフェイス９を介してＰＣ５側のＲＡＭ１３上に転送される。

【００４３】そして、ＰＣ５に対してプログラム実行用のコマンドが与えられると、ＰＣ５は、ＲＡＭ１３上に転送されているプログラムの実行コードファイル１９の内容を実行する。また、メモリ配置情報２０は、ＰＣ５側に送信された実行コードファイル１９を、ソースファイル１５のレベルに逆変換する場合に使用される。

【００４４】以上のように本実施例によれば、変換処理部７のアドレス割付実行部７ｂは、プログラム中において作業者により特定の名称が付与された自動変数が使用されていると、その自動変数を介して渡されるデータの格納領域をＰＣ５側のＲＡＭ１３上に設定されるリレー若しくはワード割付領域中に割付けるようにした。

【００４５】従って、作業者は、プログラミングを行なう際に自動変数についてアドレスの割付処理を一切意識する必要がなく、プログラムの作成が容易になり、プログラムの作成を効率良く行なうことができる。また、作業者が付与した変数名を使用することによって作成後のプログラムの理解も容易となる。

【００４６】また、本実施例によれば、自動変数のアドレスは、ＰＣ５によって実行される以前に全て確定するので、例えば、ＰＣ５のＣＰＵ１１が実行コードファイル１９の内容を実行する段階で、スタック領域を用いて自動変数のアドレスを割付けるような方式とは異なり、ＣＰＵ１１が、実行段階において一々スタックポインタを参照する必要がない。従って、プログラムの処理速度を低下させることなく高速に実行させることができる。

【００４７】更に、本実施例によれば、リレー及びワード割付領域は、自動変数を介して渡されるデータのみが格納可能に設定されているので、プログラムにおいて不用意にリレー及びワード割付領域に対応するアドレスを記述することに起因して格納されているデータを破壊してしまうことがなく、データの保護を確実に行なうことができる。

【００４８】また、リレー及びワード割付領域は、作業者が、作成するプログラムの内容に応じて必要となる容量を指定すれば、指定された容量の領域が領域設定部６ａにより確保されるので、プログラムの作成をより柔軟に行なうことができる。

【００４９】加えて、本実施例によれば、作業者は、プログラムにおいて一々自動変数のメモリ属性（ワードか、リレーか）を指定せずとも、アドレス割付実行部７ｂが命令の種類に応じてオペランドの自動変数のメモリ属性を判断してワード，リレー領域に夫々割付けを行なうので、割付領域を効率的に利用して、自動変数を使用するために必要な容量を低減することができる。

【００５０】また、本実施例によれば、アドレス割付実行部７ｂは、割付領域中に新たな自動変数の割付処理を行うための空き領域が存在しない場合には、既に割付けられている自動変数の内、その時点以降では不要となる自動変数に対応する領域に、新たな自動変数を介して渡されるデータの格納領域を割付けるので、自動変数を使用するために必要な割付領域の容量を最小限にすることができる。

【００５１】本発明は上記し且つ図面に記載した実施例にのみ限定されるものではなく、次のような変形または拡張が可能である。割付領域の容量の指定は、プログラムの記述後に行っても良く、要は、変換処理部７によりプログラムの変換を行う以前に行えば良い。また、割付領域の容量の指定は必ずしも行う必要はなく、指定がない場合はデフォルト値で設定するようにしても良い。割

付領域に空きがない場合に非競合領域に割付を行う機能は必要に応じて設ければ良く、図５に示すフローチャートのステップＳ５，Ｓ６，Ｓ９を削除して、判断ステップＳ３またはＳ８で「ＮＯ」と判断した場合は、ステップＳ１０に移行するようにしても良い。

【００５２】割付領域として確保した領域は、必ずしも当該領域に相当するアドレスを直接記述できないように設定する必要はなく、システムの仕様に応じて適宜変更して良い。自動変数のメモリ属性がリレー／ワードの何れかを判断する場合は、必ずしも命令の種類から判断する必要はなく、例えば、メモリ属性に応じて、“Ｕｄｒ”，“Ｕｄｗ”などのように、自動変数に異なる接頭語を付与して区別させても良い。また、自動変数を識別するための特定の名称は、接頭語“Ｕｄ”を付すものに限らず、例えば特定の接尾語を付しても良く、要は識別が可能であれば形式は問わない。或いは、システムの構成上、メモリマップにおいてメモリ属性を区別する必要がない場合は、自動変数を１つの割付領域に順次割付けるようにしても良い。

【図面の簡単な説明】

【図１】本発明の一実施例における変換処理部によるユーザプログラムの変換処理を概念的に示す図

【図２】プログラマブルコントローラのメモリマップの一例を示す図

【図３】ラダー言語によるユーザプログラムの記述例を示す図

【図４】アドレスの割付処理が実行された結果の一例を示す図

【図５】アドレス割付処理部の制御内容を示すフローチャート

【図６】非競合領域に対してアドレス割付を行う処理を説明するための図１相当図

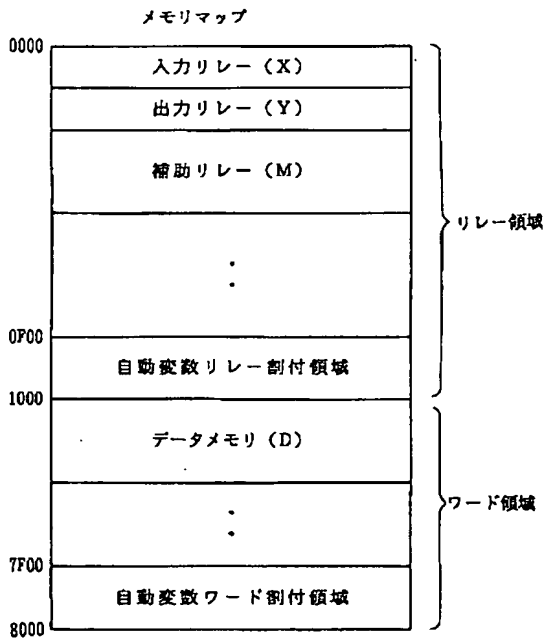
【図７】全体の電気的構成を示す機能ブロック図

【図８】プログラミング装置の詳細な構成を示す機能ブロック図

【符号の説明】

１はプログラミング装置、５はプログラマブルコントローラ、６ａは領域設定部（領域設定手段）、７は変換処理部（プログラム変換手段）、７ｂはアドレス割付実行部（割付手段）を示す。

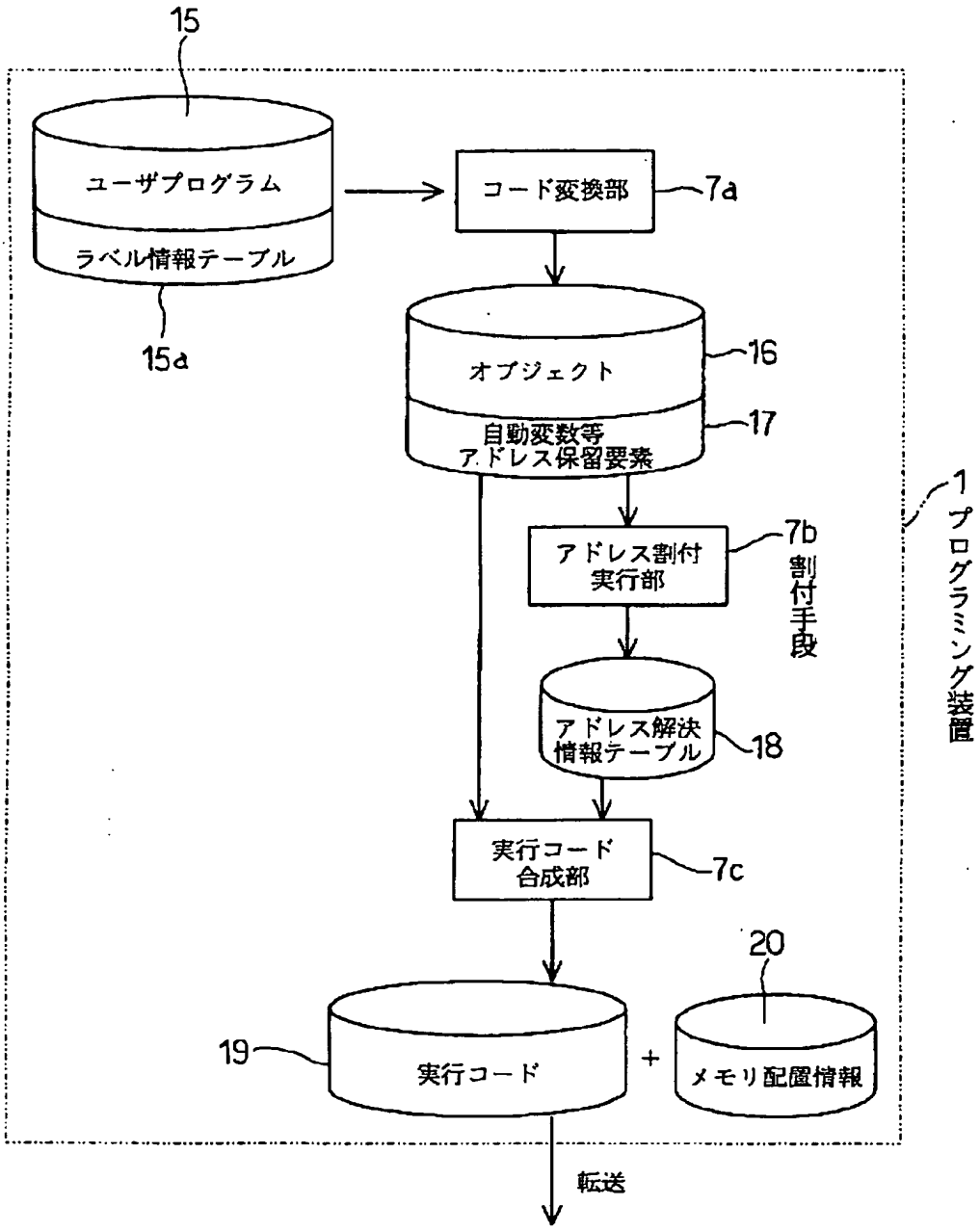
【図２】



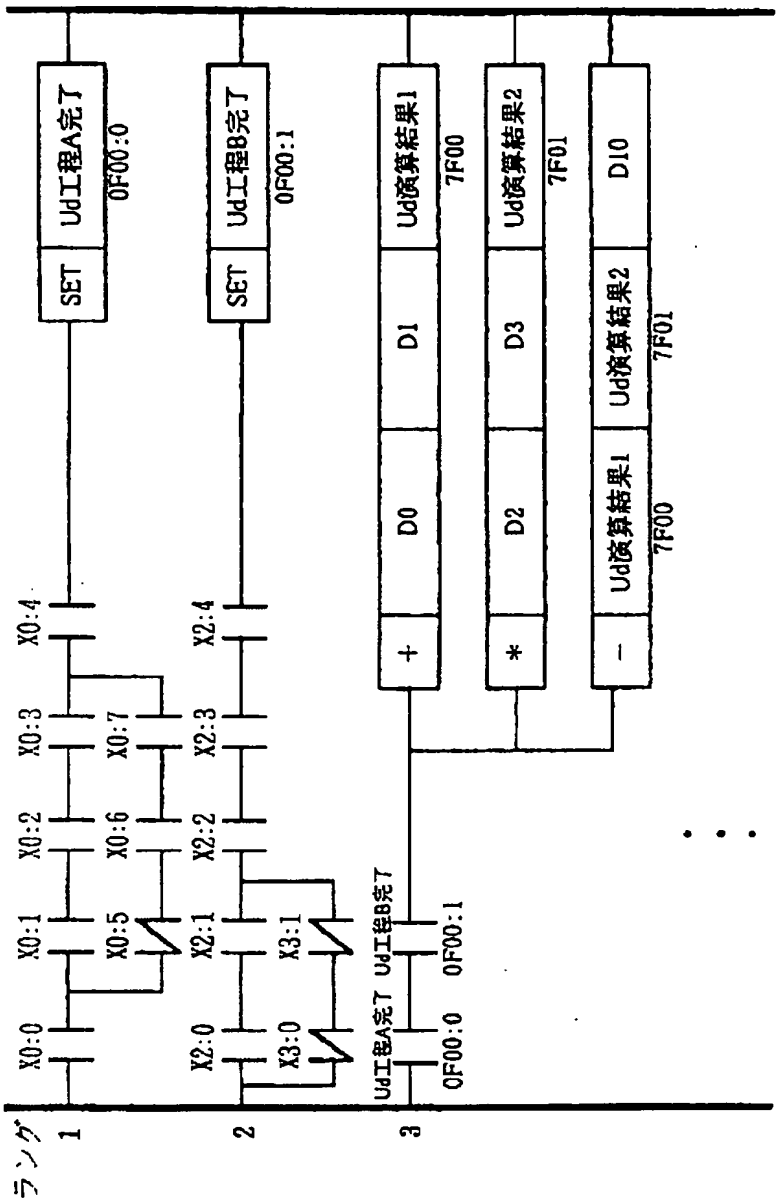
【図４】

	P1	P2	P3
SET	0F00:0(Udリレー領域A)		
SET	0F00:1(Udリレー領域B)		
+	1000(D0)	1001(D1)	7F00(Ud演算結果1)
*	1002(D2)	1003(D3)	7F01(Ud演算結果2)
−	7F00(Ud演算結果1)	7F01(Ud演算結果2)	100A(D10)

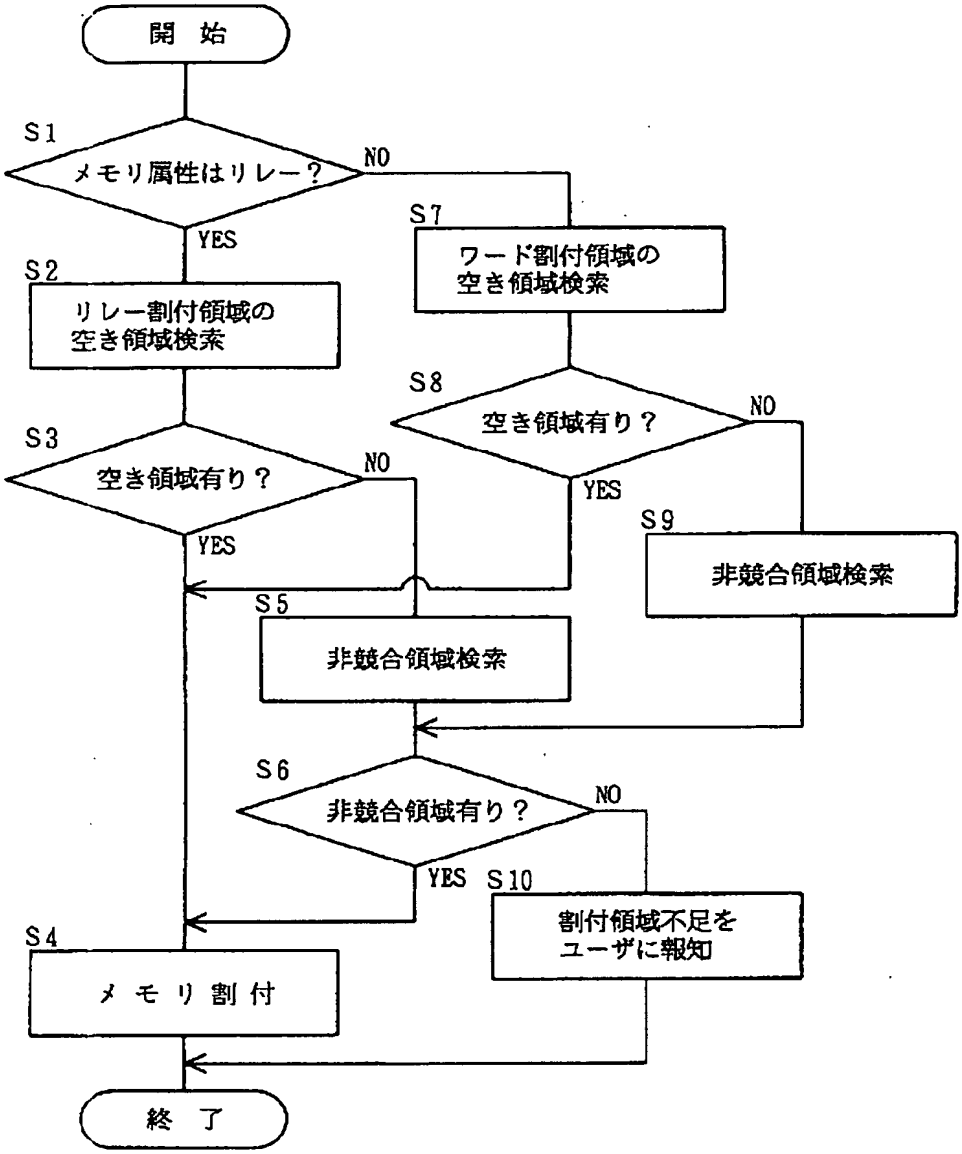
【図１】



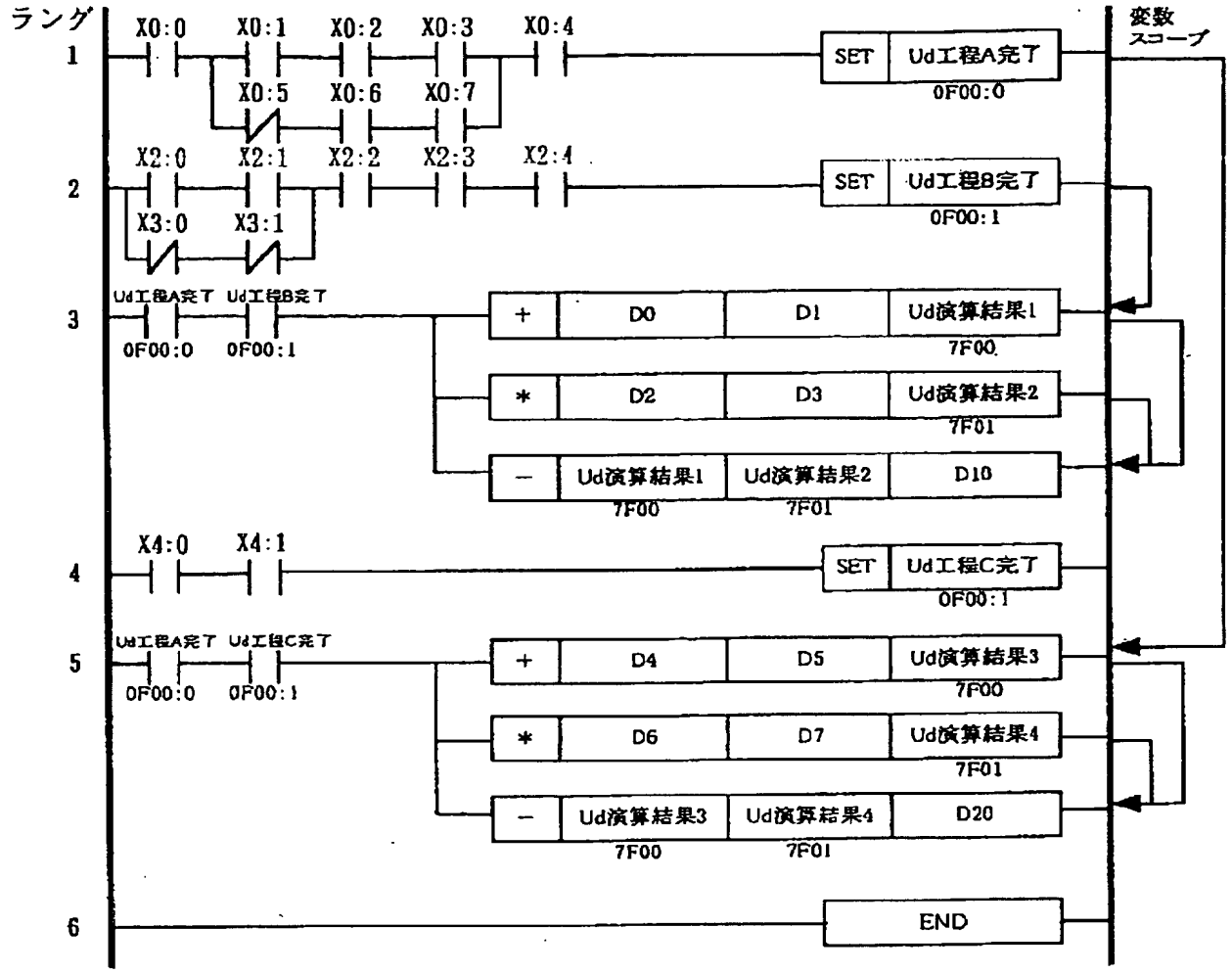
【図3】



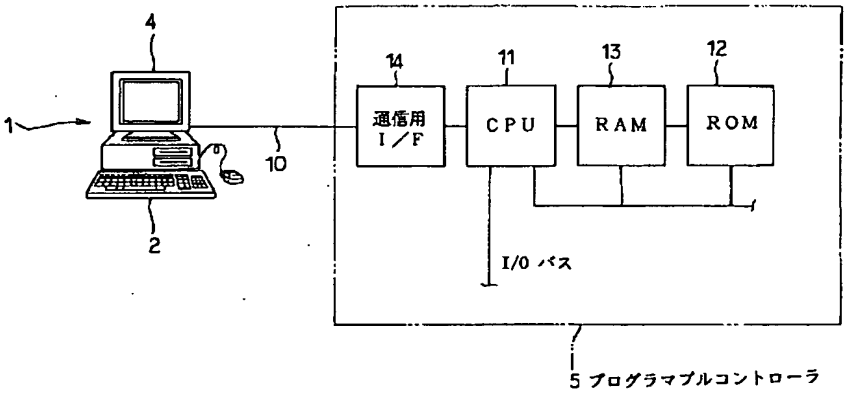
【図５】



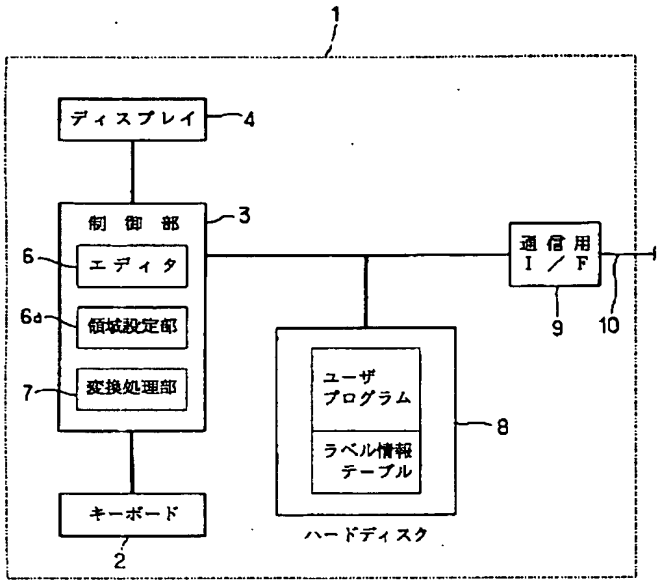
【図6】



【図7】



【図8】



6a：領域設定手段
7：プログラム変換手段